Node count problems in Graph Diffusion Models

Matteo Ninniri

Pesaresi seminars

Table of Contents

Generative models

- Learn the distribution $p(\boldsymbol{data})$ of the dataset

Generative models

- Learn the distribution $p(\boldsymbol{data})$ of the dataset
- Learn how to generate data similar to the dataset

Conditional generative models

- Learn the distribution $p(\boldsymbol{data}|\boldsymbol{y})$

Conditional generative models

- Learn the distribution $p(\boldsymbol{data}|\boldsymbol{y})$
- Learn how to generate data close to the dataset **featuring the desired input properties y.**

Conditional generative models

- Learn the distribution $p(\boldsymbol{data}|\boldsymbol{y})$
- Learn how to generate data close to the dataset **featuring the desired input properties y.**
- A textbook example: AI art

Conditional generative models

Another type of data: graphs

- 3D models
- Social interactions networks
- Molecules

Conditional generative models

Another type of data: graphs

- 3D models
- Social interactions networks
- Molecules
  $\Rightarrow$ Personalized medicine?

$\boldsymbol{y}$: *"A molecule with antimalarial properties"*
Output:

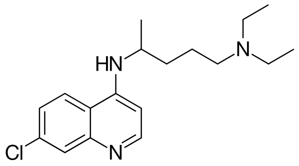
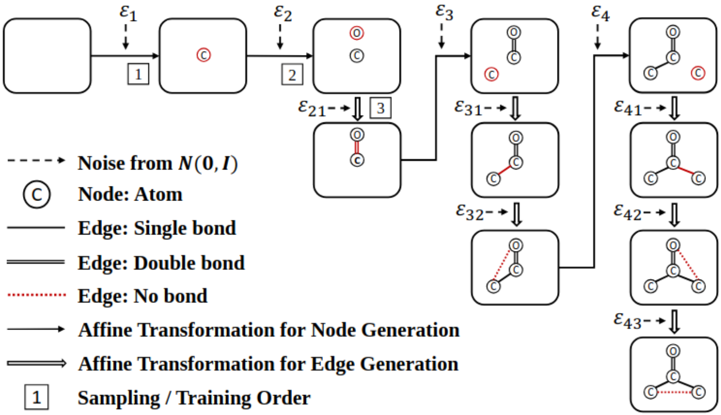


Table of Contents

Autoregressive models

Generate the sample one node/edge at a time



One-shot generative models

Generate the sample in one single step:

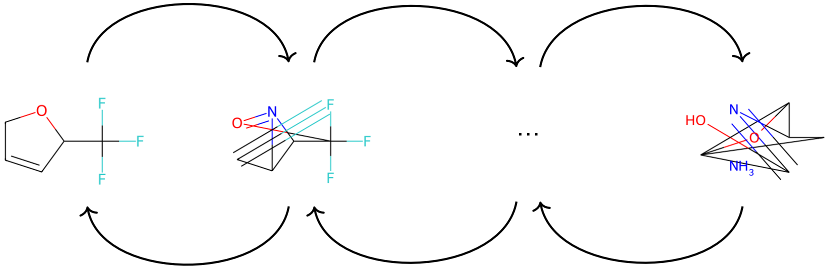
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
- Denoising Diffusion Probabilistic Models (DDPM)

Denoising Diffusion Probabilistic Models

An interesting type of "one-shot" model:

Input: graph $G_0 = (\boldsymbol{X}_0, \boldsymbol{E}_0)$

$\boldsymbol{G}_1 \sim p(\boldsymbol{G}_1 | \boldsymbol{G}_0)$ $\dots$ $\boldsymbol{G}_T \sim p(\boldsymbol{G}_T | \boldsymbol{G}_{T-1})$



$(\boldsymbol{X}_0, \boldsymbol{E}_0) \sim p_\theta(G^0 | G^1)$ $\dots$ $(\boldsymbol{X}_{T-1}, \boldsymbol{E}_{T-1}) \sim p_\theta(G^{T-1} | G^T)$

Diffusion process (discrete case)

Each node and edge in $\boldsymbol{G}_0$ is corrupted independently using

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \boldsymbol{x}_{t-1}^{'}\boldsymbol{Q}_t$$

Diffusion process (discrete case)

$[\boldsymbol{Q}_t]_{i,j} = p(x_t = i | x_{t-1} = j)$

Traditionally, $\boldsymbol{Q}_t$ is the following convex combination:

$$\boldsymbol{Q}_t = \alpha_t I + (1 - \alpha_t)(\mathbb{1}\boldsymbol{m}')$$

where

- $\boldsymbol{m}$ = marginal distribution of the node/edge categories
- α_t is a *noise scheduler*
  - $\alpha_1 = 1$
  - $\alpha_T = 0$

Denoising process (discrete case)

$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \sum_{x\in\chi} p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0 = \boldsymbol{x}) p_\theta(\boldsymbol{x}_0 = \boldsymbol{x}|\boldsymbol{x}_t)$

- $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0 = \boldsymbol{x})$ can be computed directly
- $p_\theta(\boldsymbol{x}_0 = \boldsymbol{x}|\boldsymbol{x}_t)$ is inferred through a neural network

Table of Contents

How many nodes should the graph have?

DDPMs fix the number of nodes n before denoising.

- $n \sim p(n)$ with $p(n)$ computed using the training set

This is problematic in conditional generation

- If $\boldsymbol{y}$ correlates with n, the generation may fail
  $\Rightarrow$ e.g. fewer atoms correlate with a low molecular weight

Possible solution

Solution

- Ninniri et. al ([NPB24]): train a separate model $p_\varepsilon(n \mid \boldsymbol{y})$ to compute $p(n|\boldsymbol{y})$
- Use it to choose n before starting the generative process

Problem number 2

Molecular optimization: change $\boldsymbol{y}$ editing $\boldsymbol{G}_0$ as little as possible.

"Logical" approach in DDPM:

- Corrupt the molecule up to a certain step $t < T$
- Change the input $\boldsymbol{y}$
- Denoise

$\textcircled{!}$ what if changing $\boldsymbol{y}$ requires a different amount of nodes?

Problem number 2

- Ketata et. al ([KGS+24]): corrupt $\boldsymbol{G}_0$ as $\boldsymbol{G}_{\frac{T}{2}}$, add nodes, and denoise.
  $\textcircled{!}$ but what if we need *less* nodes?

What if we want to solve both problems at the same time?

Table of Contents

Insert and delete in DDPMs

DDPMs generate a graph through an iterative process: can we add or remove items during it?

(!) Not trivial:

- What value should we give to the inserted nodes?
- Which nodes should we delete?

Insert and delete in DDPMs

General idea:

- Choose the graph size at step T, "n_T"
- Gradually insert/delete nodes until we reach the target size

Advantages:

- Very fast algorithm
- Full control over the final number of nodes
- Does not depend on T

Deletions

Let's start with the deletions.

- General idea: treat a deletion as a category itself

$$\boldsymbol{Q}_t = \begin{array}{c|ccccc|} & & & & DEL & DEL^* \\ & & & & 0 & p(del) \\ & & \text{"old" } \boldsymbol{Q}_t & & \vdots & \vdots \\ & & & & 0 & p(del) \\ DEL & 0 & \dots & 0 & 1 & 0 \\ DEL^* & 0 & \dots & 0 & 1 & 0 \end{array}$$

Deletions

Issues with this formulation:

- $\boldsymbol{G}_T$ would have delete in it
  $\Rightarrow$ $\boldsymbol{m}$ is not the marginal distribution anymore
- *Every* item can be deleted
  $\Rightarrow$ We want only $n_T - n_0$ items deleted
- No item is guaranteed to be deleted
  $\Rightarrow$ We want *exactly* $n_T - n_0$ items deleted

Deletions

Solution: hybrid diffusion

- nT elements are corrupted as before
- $n_0 - n_T$ elements are corrupted using the following $\boldsymbol{Q}_t$ matrix:

$$\boldsymbol{Q}_t = z_t(\alpha_t \boldsymbol{I} + (1 - \overline{\alpha}_t)(\mathbb{1}\boldsymbol{m}^{'})) + (1 - z_t)\boldsymbol{C} \qquad (1)$$

where z_t is a *delete scheduler* such that $z_1 = 1, z_T = 0$, and

$$\boldsymbol{C} = \begin{array}{c} \\ \\ \\ \\ DEL \\ DEL_t \end{array} \begin{pmatrix} & & & DEL & DEL_t \\ & & & 0 & 1 \\ & 0 & & \vdots & \vdots \\ & & & 0 & 1 \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}$$

Deletions

During denoising, we must re-insert the deleted nodes

- Train a separate neural network $p_\phi(n_t|G_t)$ to predict how many "DEL^*" we have to re-insert
- Insert $p_\phi(n_t|G_t)$ "DEL^*" nodes in G_t
- Compute $p(G_{t-1}|G_t)$ as before

Inserted node value

What category should a new node have once inserted?

Many possibilities:

- Random

Inserted node value

What category should a new node have once inserted?

Many possibilities:

- Random
  $\textcircled{!}$ not informative when t is small

Inserted node value

What category should a new node have once inserted?

Many possibilities:

- Random
  $\textcircled{!}$ not informative when t is small
- Use a neural network

Inserted node value

What category should a new node have once inserted?

Many possibilities:

- Random
  (!) not informative when t is small
- Use a neural network
  (!) Too complex

Inserted node value

What category should a new node have once inserted?

Many possibilities:

- Random
  ! not informative when t is small
- Use a neural network
  ! Too complex
- Sample from $\boldsymbol{m_x}$

Inserted node value

What category should a new node have once inserted?

Many possibilities:

- Random
  ⓘ! not informative when t is small
- Use a neural network
  ! Too complex
- Sample from $\boldsymbol{m_x}$
  ! duplicating rather than inserting

Inserted node value

What category should a new node have once inserted?

Many possibilities:

- Random
  ! not informative when t is small
- Use a neural network
  ! Too complex
- Sample from $\boldsymbol{m_x}$
  ! duplicating rather than inserting

In conclusion, it's still an open problem

Inserted node value

When denoising, we delete the nodes that were inserted during the diffusion process

$\textcircled{!}$ in standard DDPMs, we use $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)$. But a node inserted during the diffusion process does not exist at $t = 0$!

Inserted node value

Solution: predict the next best thing: $\boldsymbol{x}$ at the insertion step i

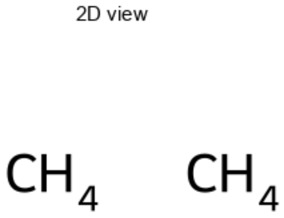
$\textcircled{!}$ p_θ needs to predict the insertion step as well!

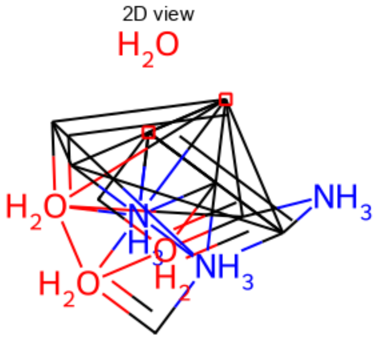
$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \sum_{x\in\chi} p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_i = \boldsymbol{x})p_\theta(\boldsymbol{x}_i = \boldsymbol{x}|\boldsymbol{x}_t) \tag{2}$$

$$\tag{3}$$

Table of Contents

(Preliminary) results





Comclusions

Can insertions and deletions fully replace standard DDPMs? **No**

- Conditional generation lacks a little bit;
- Out-of-distribution sampling does not perform well against "standard" DDPMs;

Other applications

Molecular inpainting: Fix a molecular substructure and generate a new molecule featuring it

- Corrupt the graph to step $t < T$
- At each denoising step, place the substructure' graph in $\boldsymbol{G}_t$

Discrete data generation

- Who says that all of this only applies to graphs?

That was all.

Thank you for your attention!

Questions?

References.

- Mohamed Amine Ketata, Nicholas Gao, Johanna Sommer, Tom Wollschläger, and Stephan Günnemann, *Lift your molecules: Molecular graph generation in latent euclidean space*, 2024.
- Matteo Ninniri, Marco Podda, and Davide Bacciu, *Classifier-free graph diffusion for molecular property targeting*, 2024.